Python과 Git으로 만드는 모바일 게임 패치 시스템

데브시스터즈 | 오영택

발표자 소개

오영택

26, 남

현 DEVSISTERS 서버팀
라이브 서비스 서버파트 및 인프라 총괄

KAIST 전산동아리 SPARCS

쿠키런

꽤 많은 분들이 아시리라 생각하는 그 모바일 게임

출시 2년 5개월차, 새로운 모습을 보이기 위해 달리는 중!

게임 개발의 목표

게임 개발의 목표

게임 개발의 목표

이경한 저건 여러분의 크리스탈값입니다.
좋아요 · 답글 달기 · 509 · 2014년 10월 24일 오후 9:10

오보배 ㅋㅋㅋㅋㅋㅋㅋㅋㅋㅋㅋㅋㅋ
좋아요 · 답글 달기 · 2014년 10월 24일 오후 11:25

남선화 김환규 김정민 ㅋㅋㅋㅋㅋㅋㅋㅋㅋㅋㅋㅋㅋㅋ
좋아요 · 답글 달기 · 2014년 10월 25일 오전 2:35

신혜인 서다찬 ㅋㅋㅋㅋㅋㅋㅋ
좋아요 · 답글 달기 · 2014년 10월 25일 오전 9:41

~~밥먹을 돈을 번다~~

게임 개발의 목표

게임은 수많은 콘텐츠의 집대성

사용자가 즐길 수 있도록

콘텐츠를 효율적으로 전달하는 것

콘텐츠의 전달의 핵심

아트웍, 사운드, UI, 밸런스 등등...

각각의 콘텐츠가 준비돼 하나로 결합하는 과정

예전의 콘텐츠 준비 과정

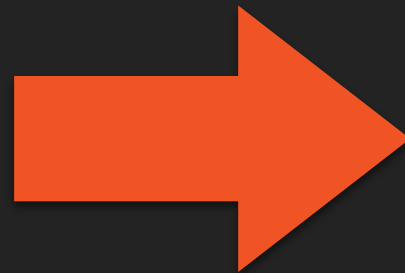
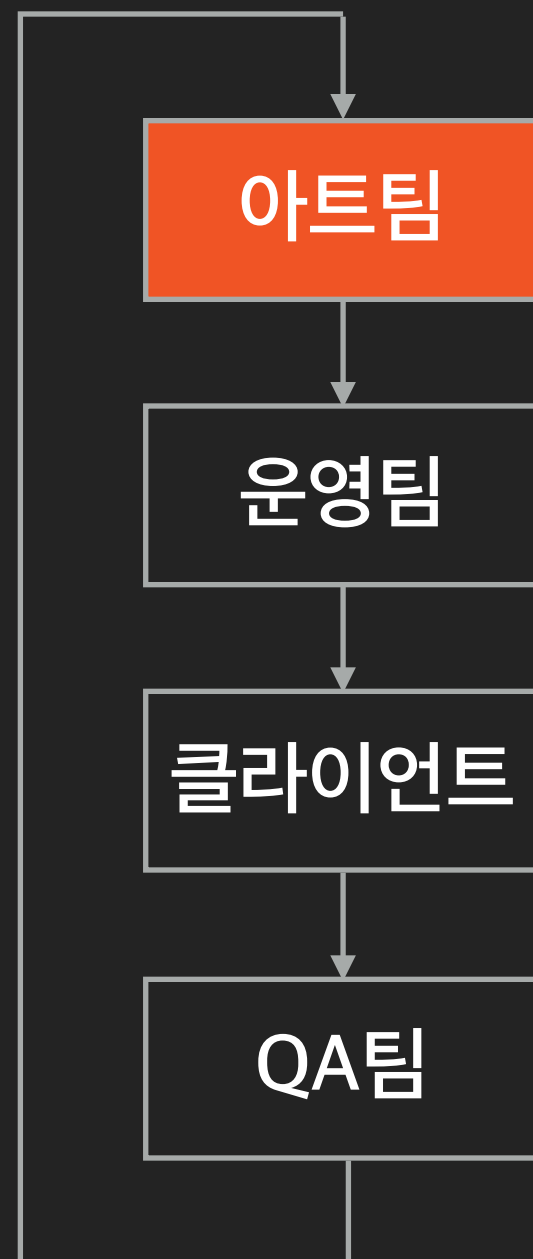
게임 패치란?

새로운 이벤트가 열릴 때, 데이터가 수정됐을 때
받는 이것

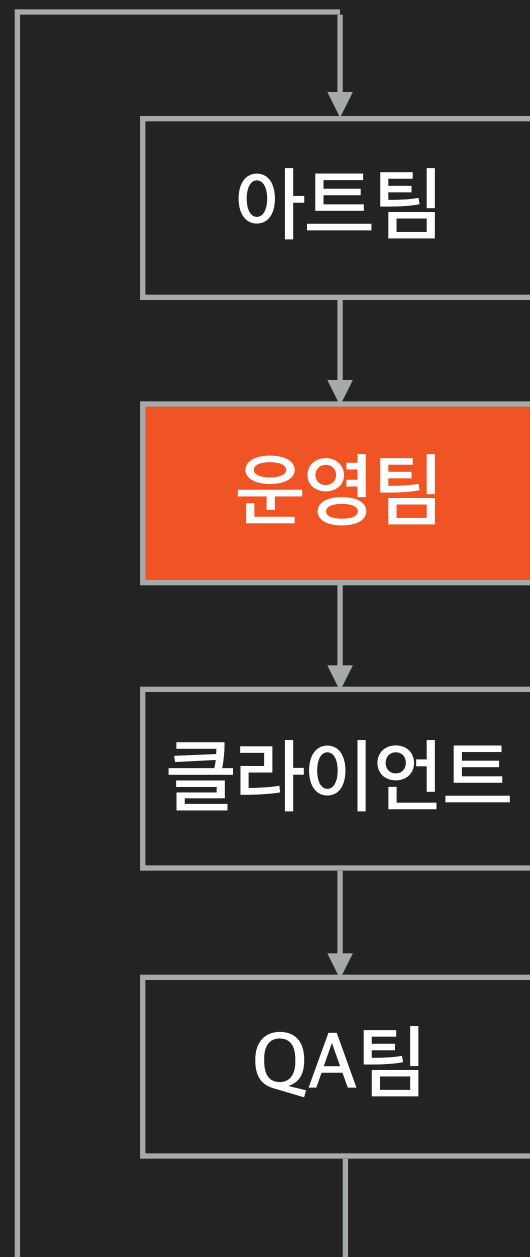
패치 시스템 구조

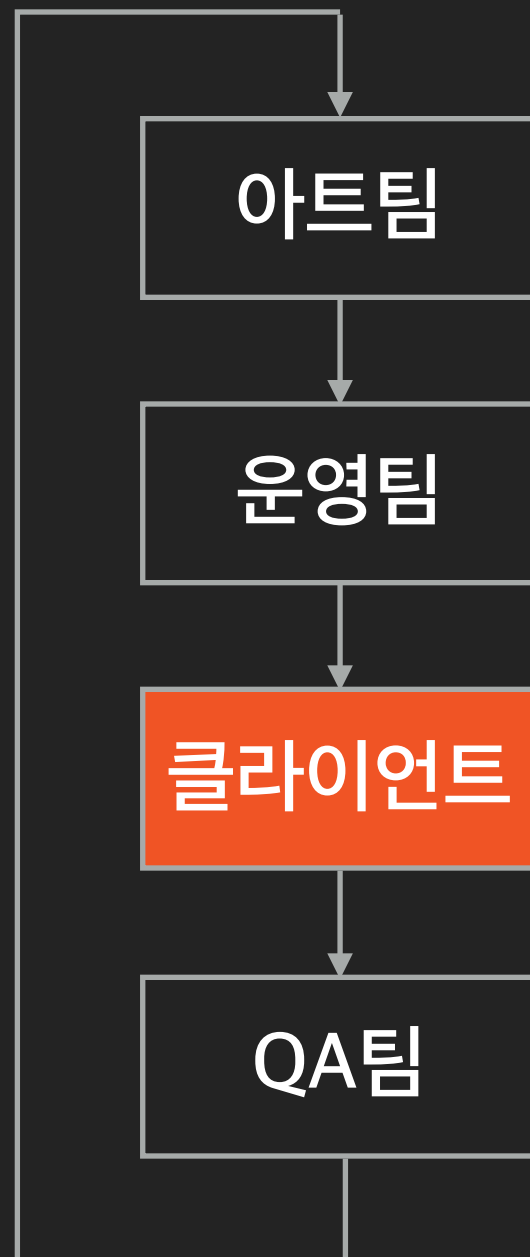
한 묶음의 콘텐츠를 가지고 있는 패치 서버

테스트가 필요한 파일을 패치 서버에 올림

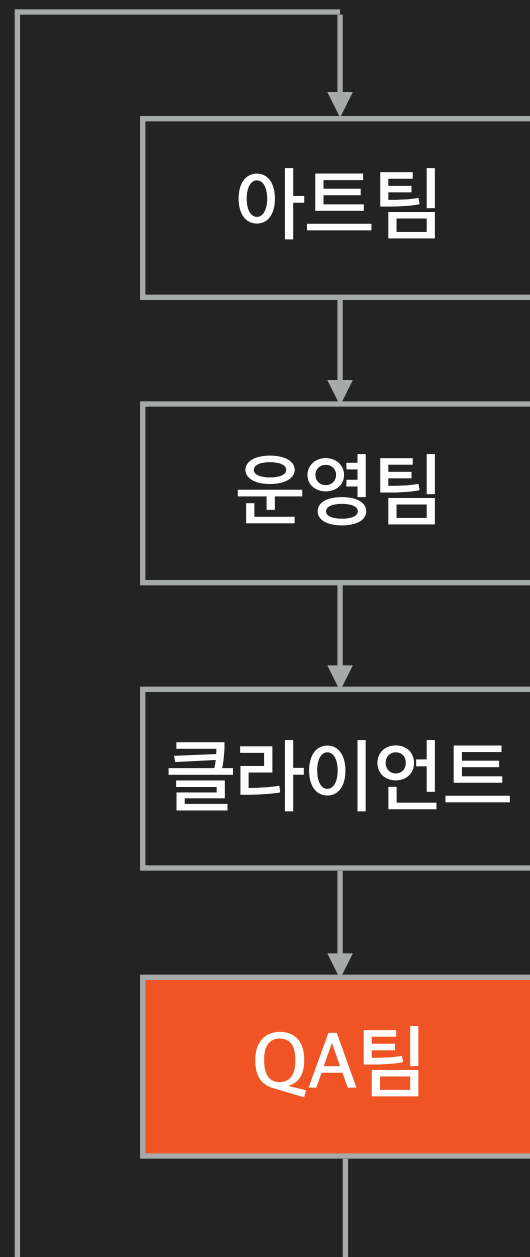
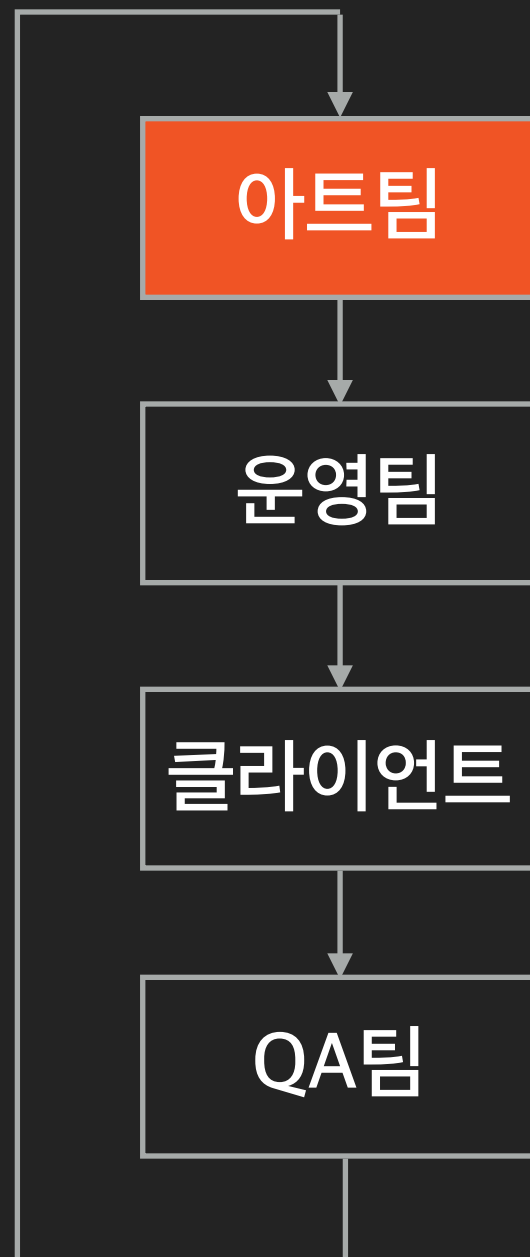
클라이언트가 패치를 다운로드 받아 테스트

쿠키 굽는 과정 (Before)

1. 아트팀이 정글전사 쿠키를 그린다

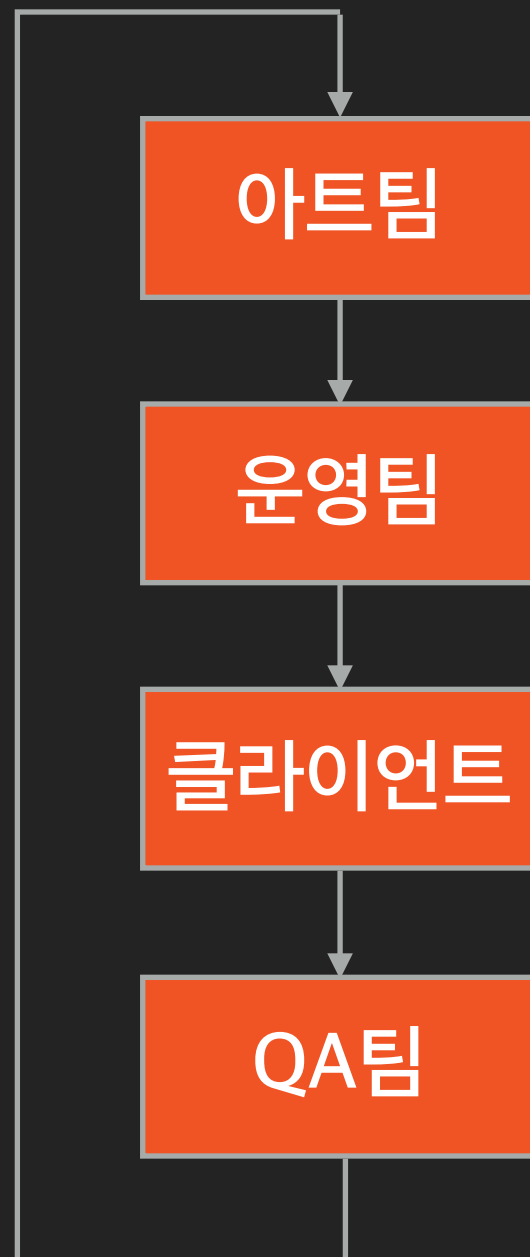
쿠키 굽는 과정 (Before)

2. 운영팀이 해당 내용을 받아서 서버에 올려본다.

쿠키 굽는 과정 (Before)

3. 올렸는데 반영이 되지 않는다. 왜인지 모르겠다.

4. 클라이언트팀에 요청해 디버그 빌드를 뽑는다.

쿠키 굽는 과정 (Before)

쿠키 굽는 과정 (Before)

6. 고친다. 나. 정글전사

쿠키 굽는 과정 (Before)

문제 확인을 위해 **거쳐야하는 팀들**?

➡ 적으면 2개, 보통 3개팀

문제를 확인하기까지 **걸리는 시간**?

➡ 최소 1~2시간, 길게는 반나절

콘텐츠를 독립적으로 확인하기 어려움

저 이 파일 업로드 해도 되나요?

제 거 테스트 중이에요!
조금만 (1시간) 기다리세요.

콘텐츠를 독립적으로 확인하기 어려움

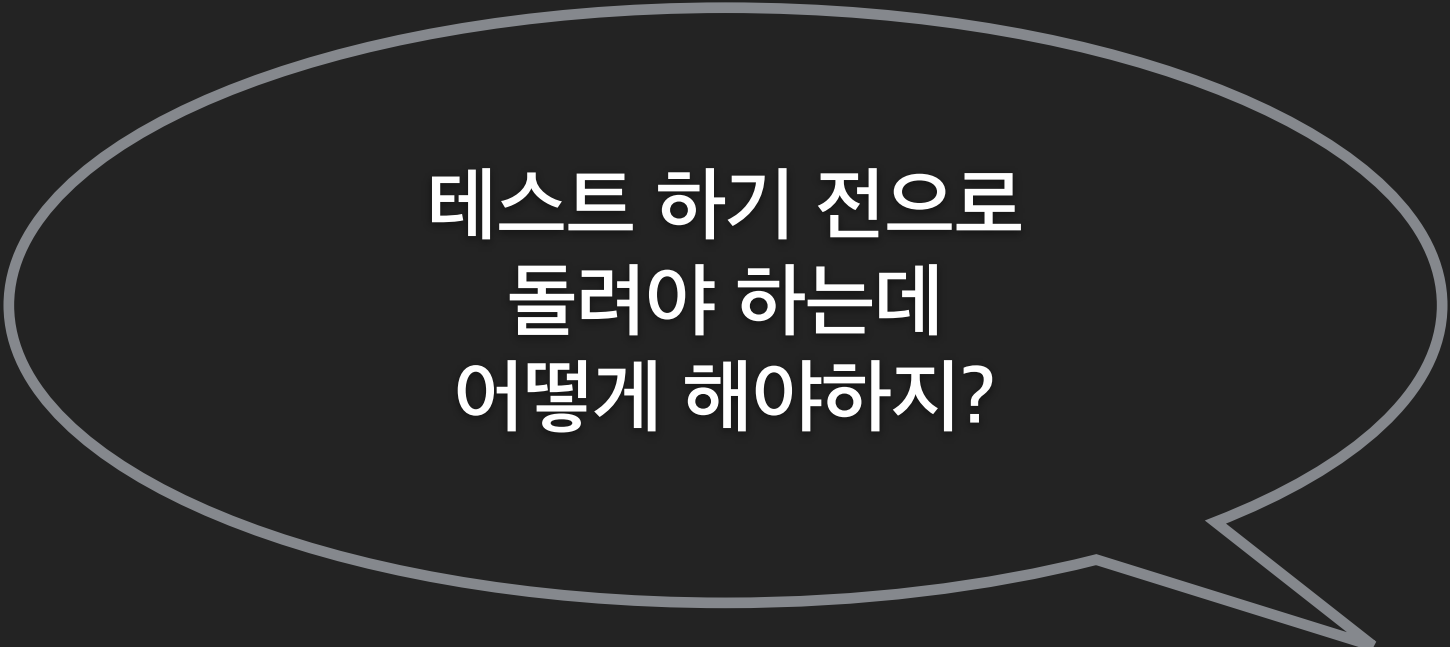
개발 환경에서 필요한만큼
독립적으로 콘텐츠 묶음을 만들 수 있어야 함

저 이 파일 업로드 해도 되나요?

제 거 테스트 중이에요!
조금만 (1시간) 기다리세요.

언제 어떤 콘텐츠를 올렸는지 알 수 없음

쿠키 이미지 고친걸
안 올렸나?

이 때 업데이트 한 게
어떤 파일들인지 모르겠어..

언제 어떤 콘텐츠를 올렸는지 알 수 없음

콘텐츠가 전부 관리되고,
변경점이 명확하게 파악되어야 함.

쿠키 이미지 고친걸
안 올렸나?

이 때 업데이트 한 게
어떤 파일들인지 모르겠어..

테스트 후 원래 상태로 돌리기 어려움

테스트 후 원래 상태로 돌리기 어려움

콘텐츠 묶음을 원하는 상태로
손쉽게 리셋할 수 있어야함

테스트 하기 전으로
돌려야 하는데
어떻게 해야하지?

바뀐적이 없는 파일이 올라가 혼동을 유발함

이번에 파일 5개 올려서
업데이트 했어요~

?!?!
바뀐 게 하나밖에 없을텐데??

바뀐적이 없는 파일이 올라가 혼동을 유발함

 최소 변경점만이 반영되고 패치로 생성되어야 함.

이번에 파일 5개 올려서
업데이트 했어요~

?!?!
바뀐 게 하나밖에 없을텐데??

필요한 것들 정리

개발 환경에서 필요한만큼 독립적으로 콘텐츠 묶음을 정의

콘텐츠가 전부 관리되고, 변경점이 명확하게 파악돼야함

콘텐츠 묶음을 원하는 상태로 손쉽게 리셋

최소 변경점만이 반영되고 패치로 생성돼야함

개발 환경에서 필요한만큼 독립적으로 ~~콘텐츠 묶음을~~브랜치를 정의

~~콘텐츠가~~파일이 전부 버전관리되고, 변경점이 명확하게 파악~~돼야함~~됨

~~콘텐츠 묶음을~~브랜치를 원하는 상태로 손쉽게 리셋

최소 변경점만이 반영되고 패치로 생성~~돼야함~~할 수 있음

Git

개발 환경에서 필요한만큼 독립적으로 브랜치를 정의

파일이 전부 버전관리되고, 변경점이 명확하게 파악됨

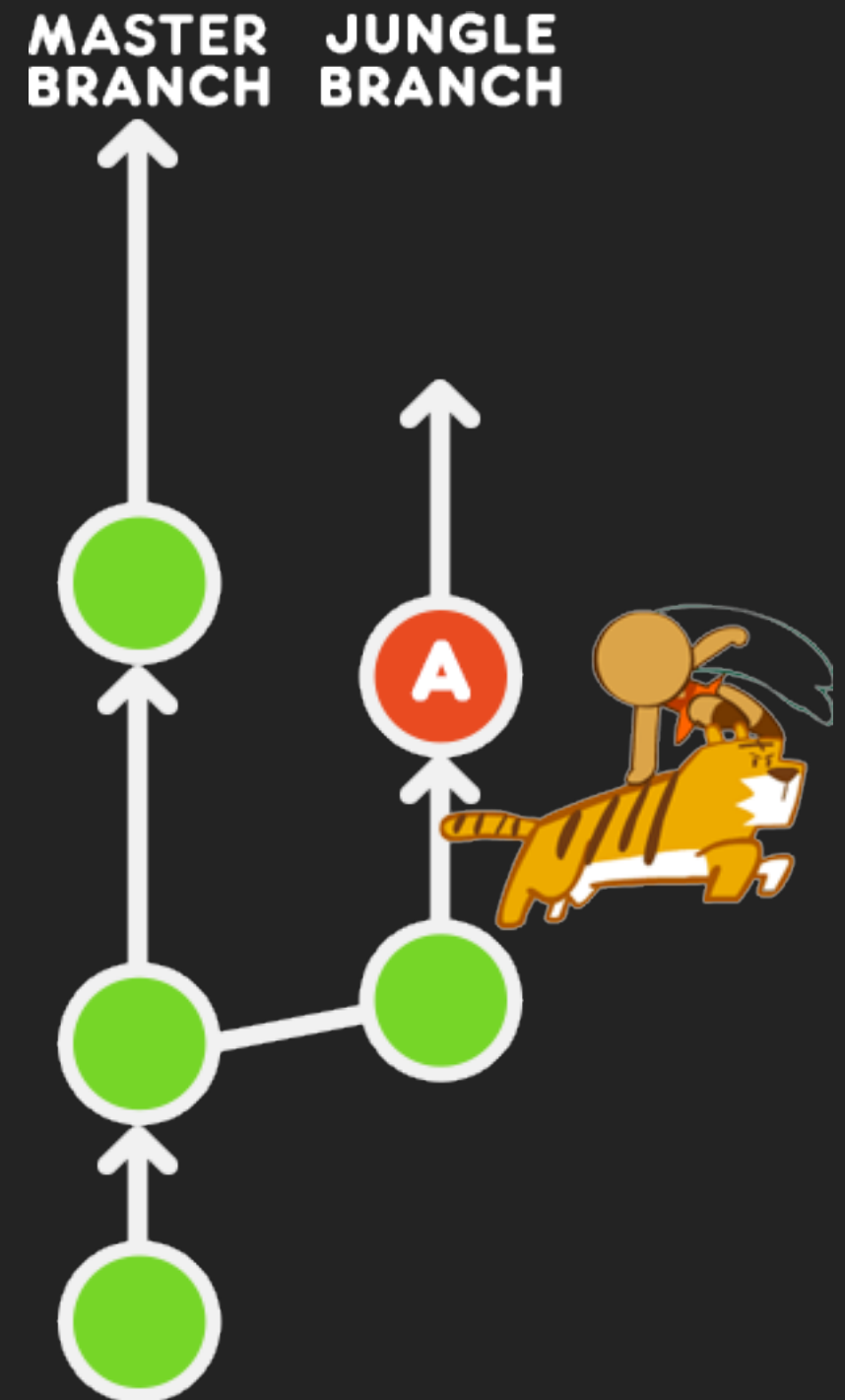
브랜치를 원하는 상태로 손쉽게 리셋

최소 변경점만이 반영되고 패치로 생성할 수 있음

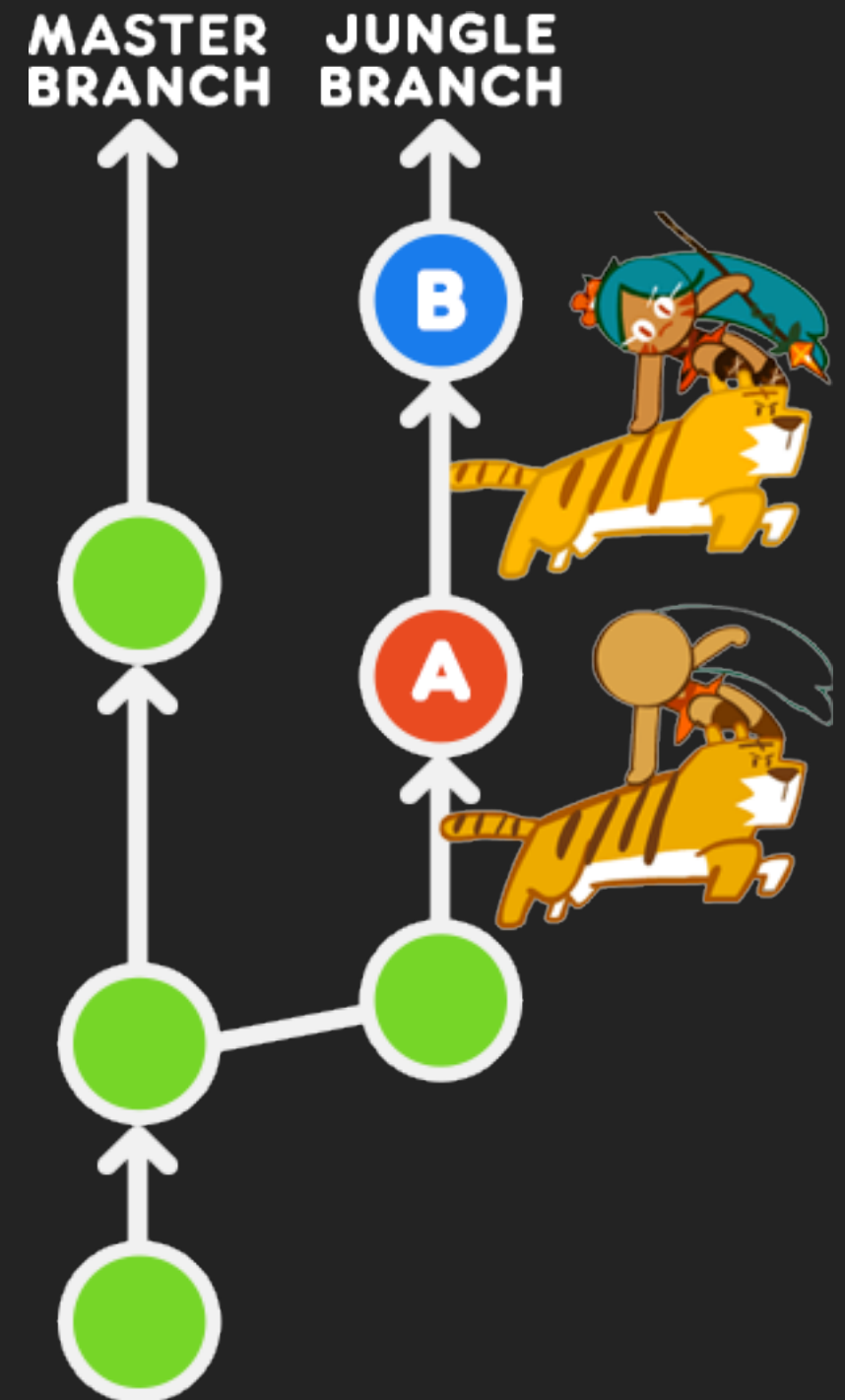
새로운 시스템의 구조

기본 작동 방식

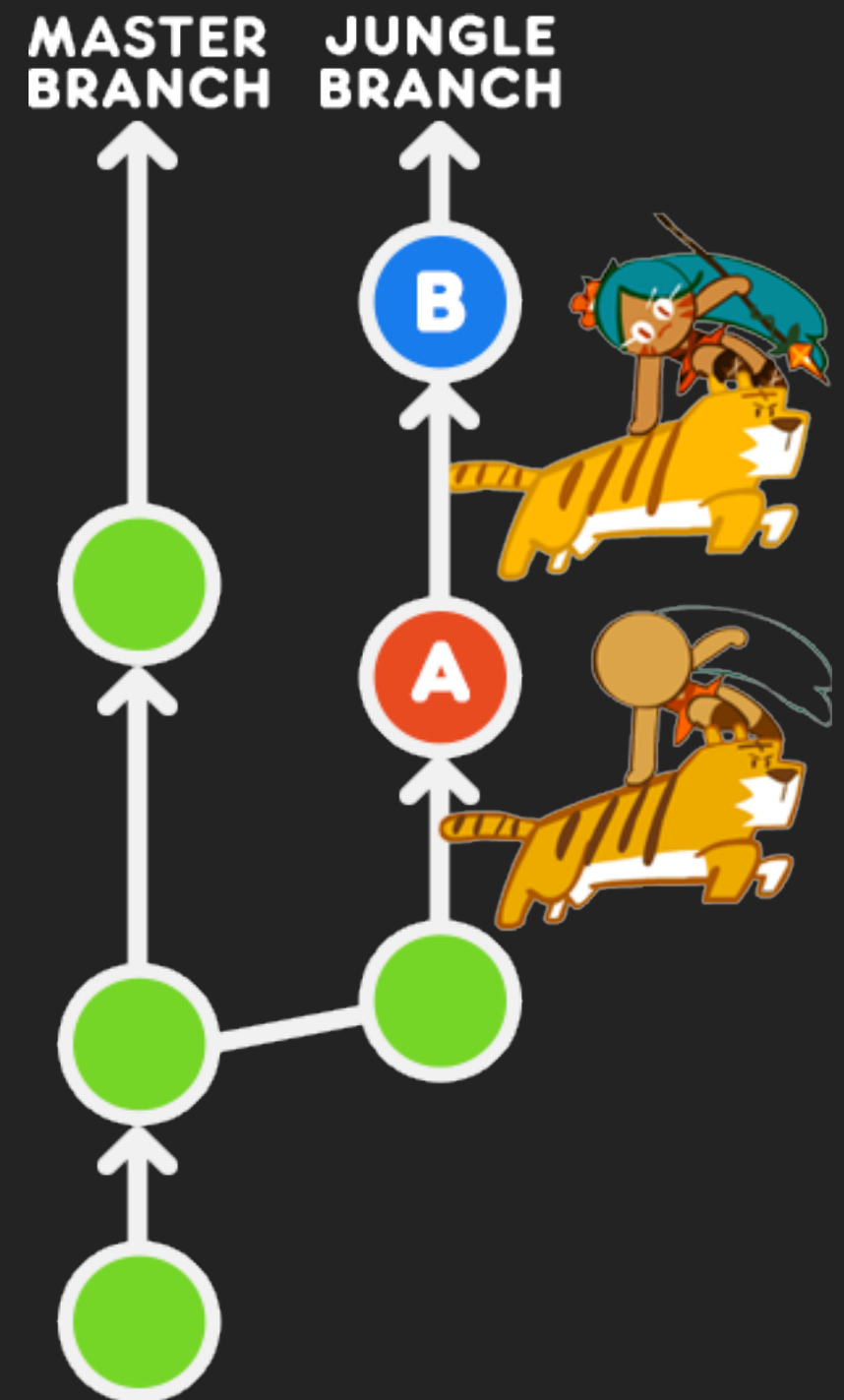
테스트 클라이언트는
Jungle 브랜치를 바라봄

테스트해보니
얼굴 없는 정글전사를 포착!(A)

기본 작동 방식

수정된 내용을 커밋(B)

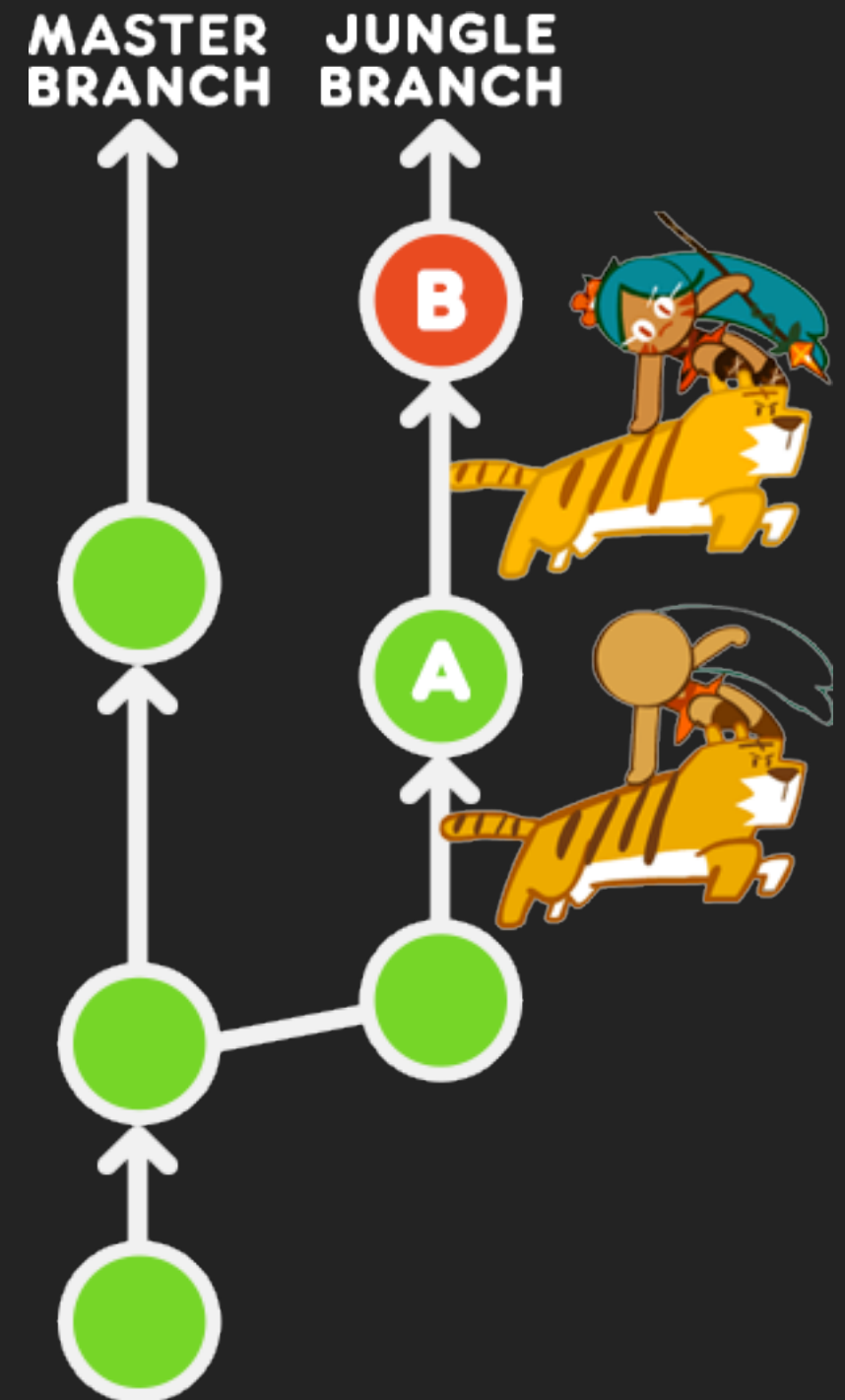
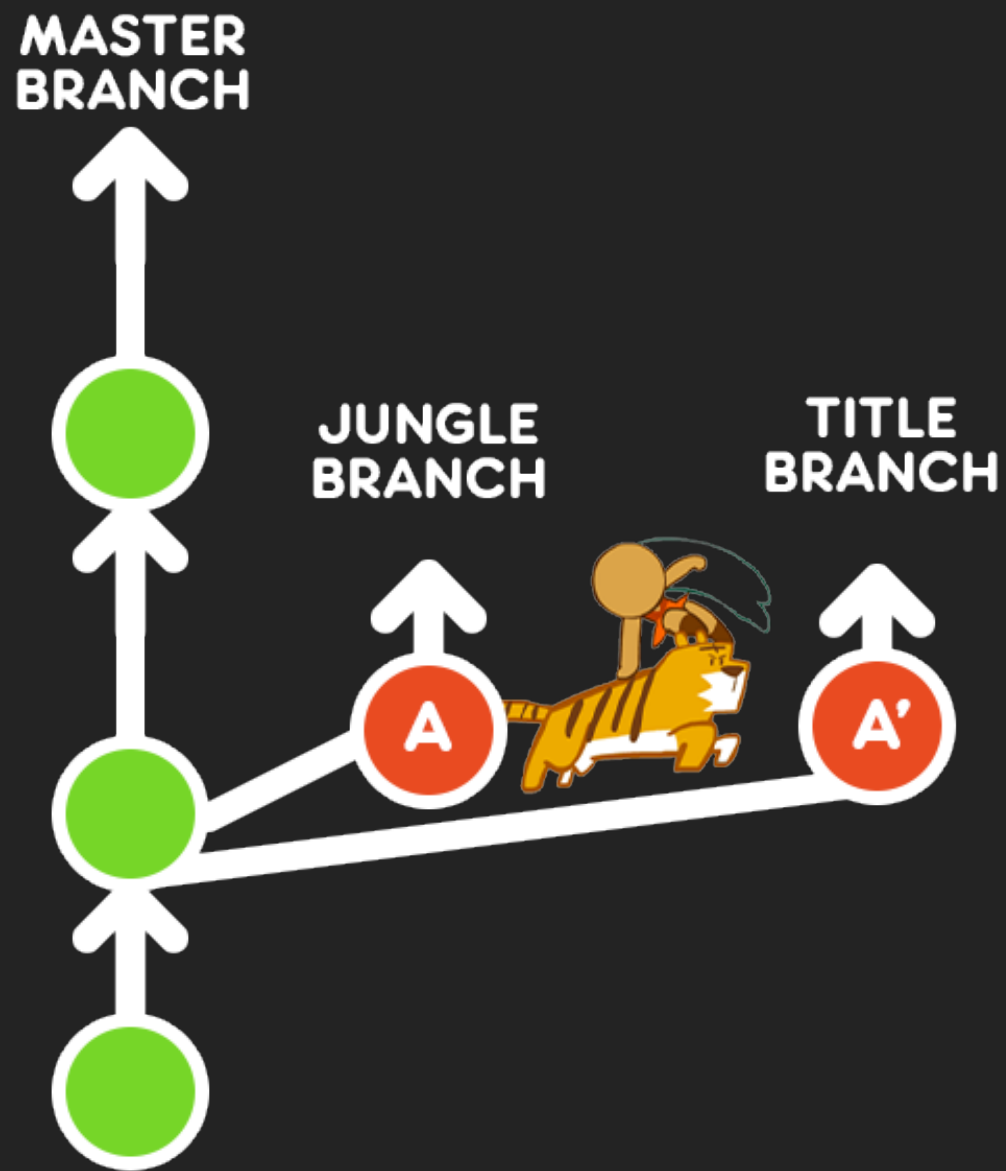
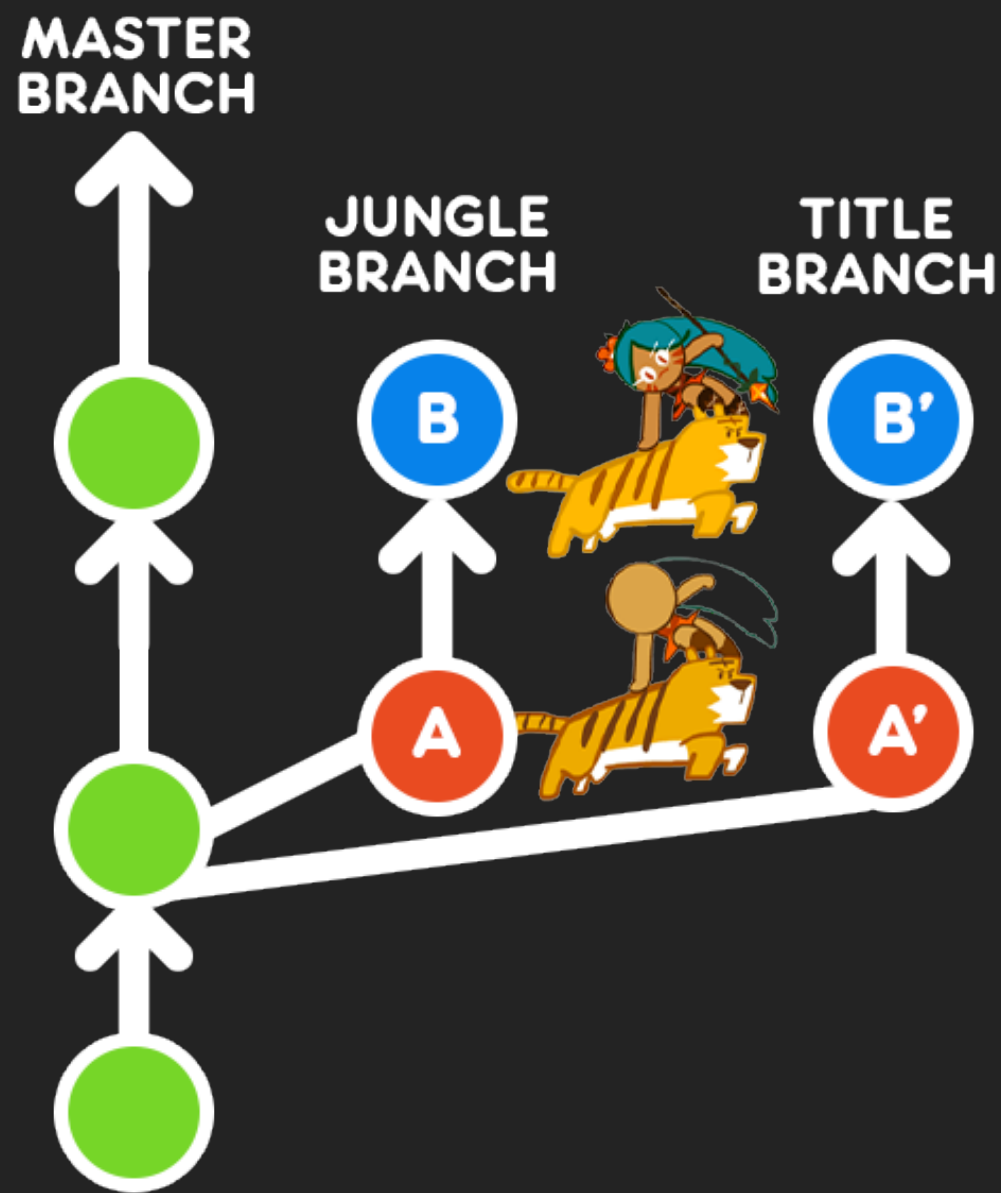
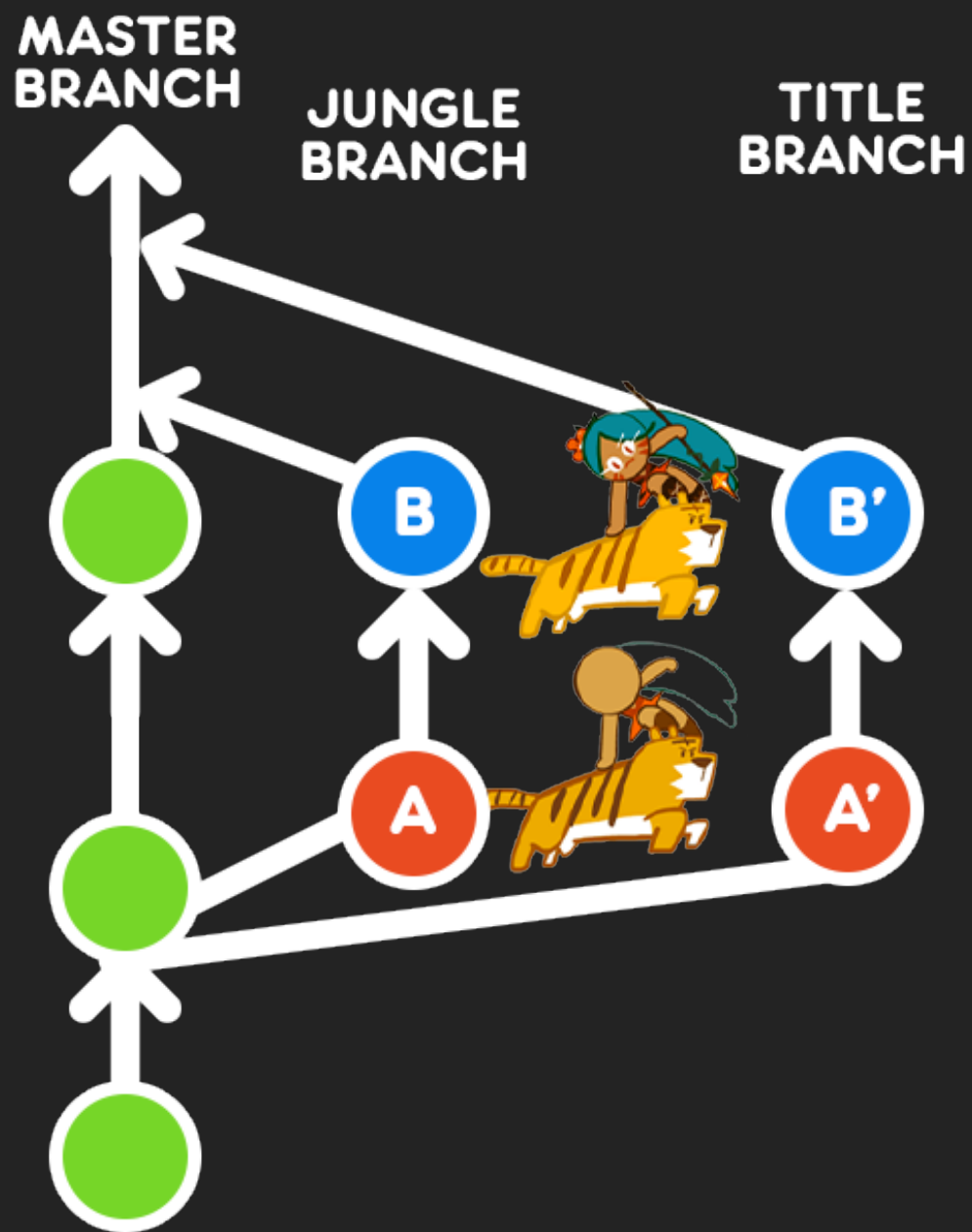
이후 클라이언트를 실행하면
새로운 데이터가 있음을 알 수 있게됨

기본 작동 방식

클라이언트는
Jungle 브랜치의 최신 커밋이
자신이 가진 A가 아닌 B임을 파악

A에서 B로 가는 패치를 서버에 요청

기본 작동 방식

서버가 내려준 패치 A to B를 받아 적용

이 패치에는 새로운 정글전사 이미지(B)가 포함됨

정글전사에 얼굴이 생겼다!

기본 작동 방식

기본 작동 방식

기본 작동 방식

만든 것 1. 패치 파일 서버

Git 원격 저장소로부터 최신 내용을 항상 받아옴

클라이언트에 브랜치의 최신 커밋을 알려줌

요청받은 커밋 A와 커밋 B 사이에
달라진 파일들을 하나로 압축해 내려줌

만든 것 2. Git 웹 클라이언트

비개발자들(아트, UI, QA 등등)을 위해 만들어짐

원하는 브랜치에 파일을 업로드하는 기능

변경사항을 커밋해주는 시스템

기존의 Django 베이스의 쿠키런 운영 툴에 구현

만든 것 3. 콘텐츠 배포 시스템

필요한 패치 파일들을 생성

AWS S3에 업로드

CDN에 해당 파일들을 반영

사용자들에게 배포

시스템을 쓰는 방법

콘텐츠를 만들고

Git 저장소의 브랜치에 콘텐츠를 올림

시스템을 쓰는 방법

올린 파일은 패치 시스템에 자동 동기화

Github webhook을 이용

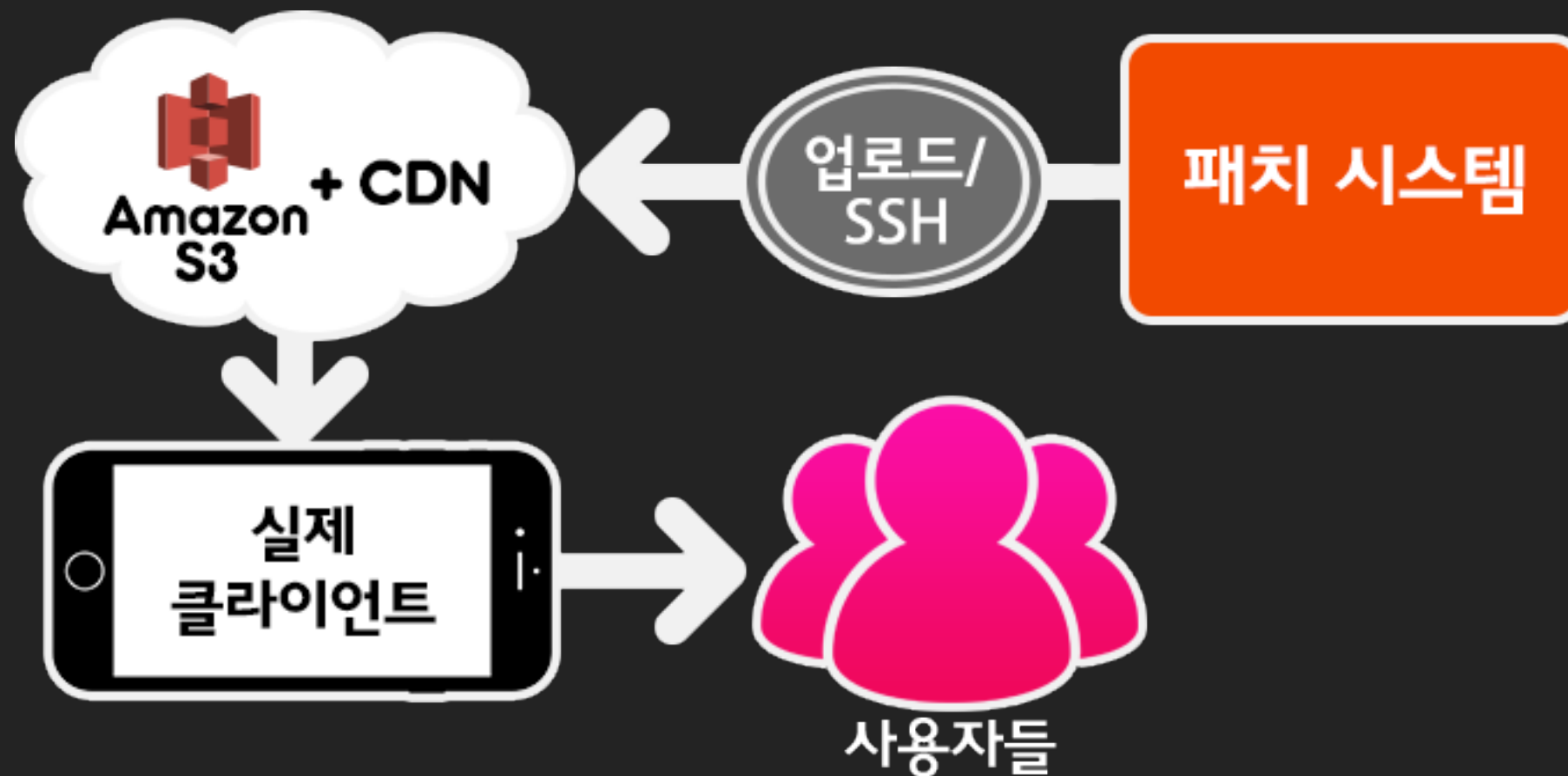
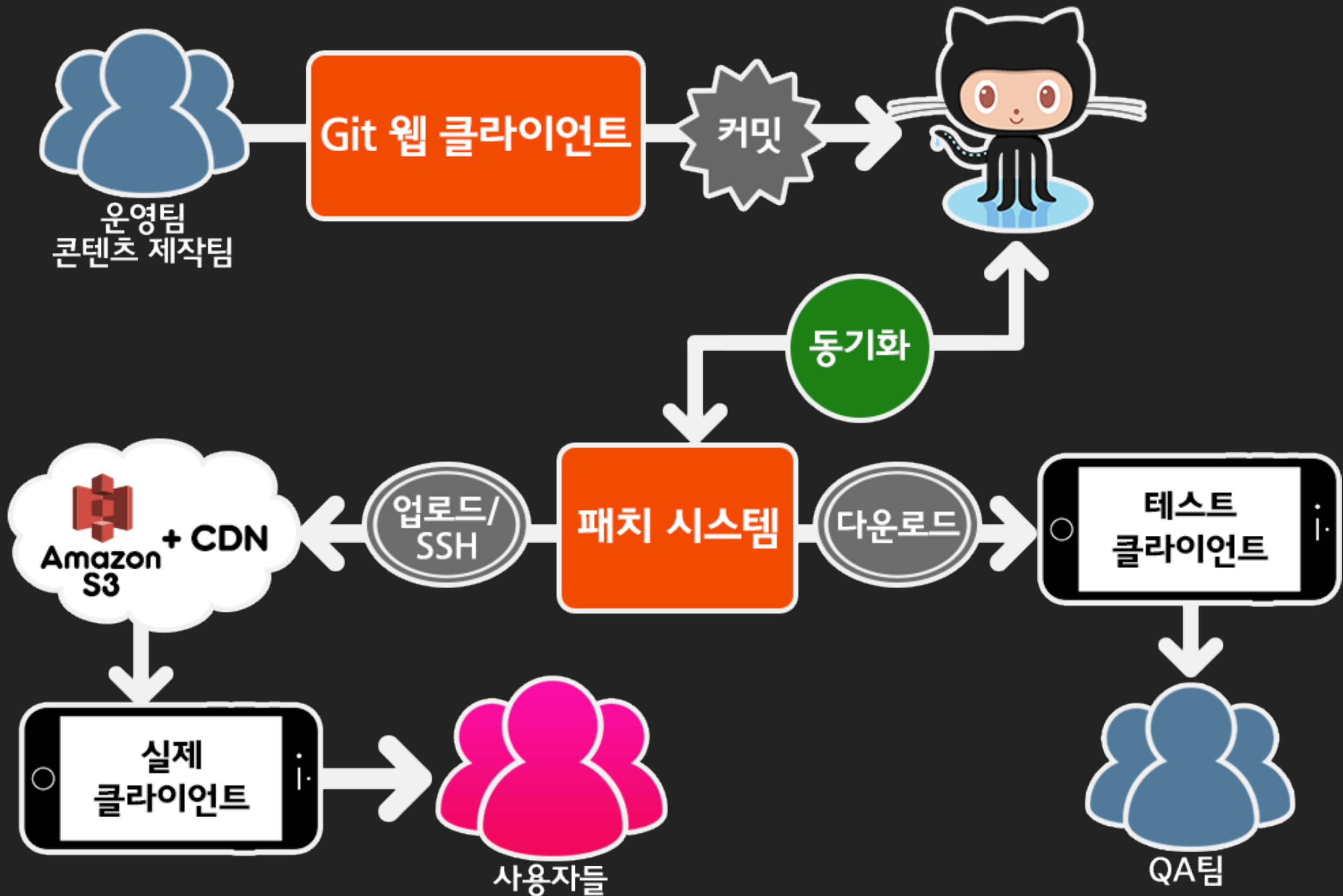
Github에서 이벤트 발생시 알려주는 기능

시스템을 쓰는 방법

테스트 클라이언트를 사용해 콘텐츠 확인

시스템을 쓰는 방법

검증이 끝난 콘텐츠는 배포 단계를 거쳐 사용자에게 전달

Git 웹 클라이언트
운영팀
콘텐츠 제작팀
커밋
동기화
Amazon S3
+ CDN
업로드/
SSH
패치 시스템
다운로드
테스트
클라이언트
실제
클라이언트
사용자들
QA팀

새로운 시스템의 재료

이걸 어떻게 만들지

Git 저장소를 손쉽게 코드로 조작

저장소를 바탕으로 필요한 파일을 생성하여 제공

콘텐츠 관리를 쉽게 만들어 줄 프론트엔드 도구

다양한 이질적인 요소들을 결합할 수 있어야 한다

Python

이질적인 요소들도 쉽게 연동할 수 있다 (glue language)

개발 속도가 빠르고, 이해하기 쉽다

라이브러리가 다양하고, 잘 알려진 것들을 믿고 쓸 수 있다

사내 많은 개발자들이 이미 파이썬에 친숙하다

Pygit2

에 기반한 라이브러리

libgit2는 github에서도 사용하고 있음

사용을 위해서는 git 내부 구조를 이해해야함
(https://git-scm.com/book/en/v1/Git-Internals 을 참고)

문서가 다소 불친절..

Pygit2

개발 당시 (2014년 중반) libgit2는 아직 안정적인 상태가 아님

어처구니 없는 버그(들)이 존재

(예 : 아무리 큰 파일도 commit push할때 첫 32KB까지만 전송하는 버그)

종종 libgit2 코드를 통해 작동 방식을 확인할 필요가 있음

Bottle

Python의 수많은 웹 프레임워크 중 하나

단일 모듈로 웹서버를 빠르고 손쉽게 구축

단순한 로직의 구현에는 매우 쓸만함

Bottle

Hello {name} 을 출력하는 웹서버 (bottlepy 첫 예제)

```python
from bottle import route, run, template

@route('/hello/<name>')
def index(name):
    return template('<b>Hello {{name}}</b>!', name=name)

run(host='localhost', port=8080)
```

Bottle

Git 저장소에서
특정 로컬 브랜치의 최신 리비전을 출력하는 웹서버

```
import bottle
import pygit2

@bottle.route('/branch/<branch_name>')
def get_latest_revision(branch_name):
    repo = pygit2.Repository('/path/to/repo')
    branch = repo.lookup_branch(branch_name)
    return str(branch.target) if branch else 'branch not found'

bottle.run(host='localhost', port=8080)
```

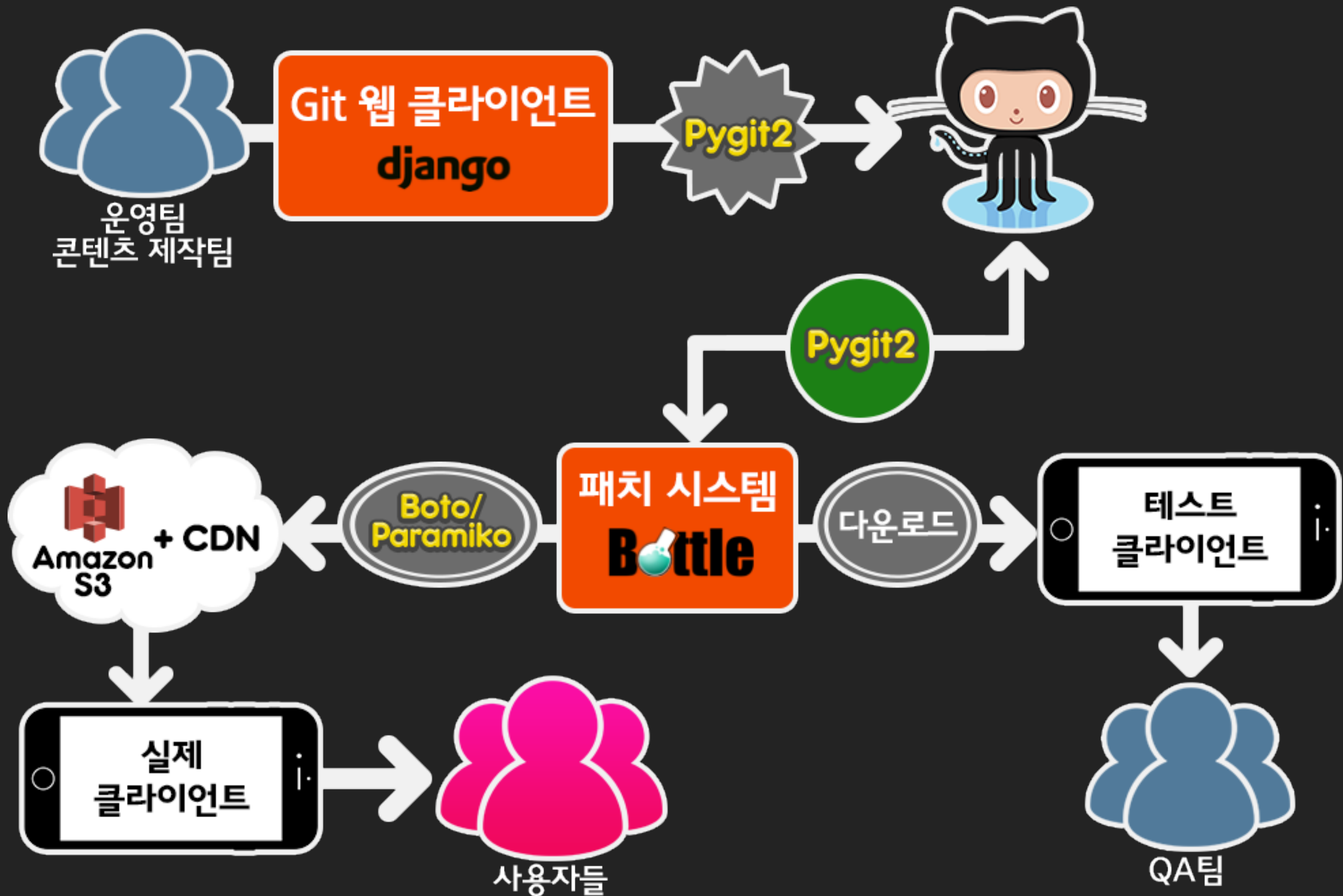
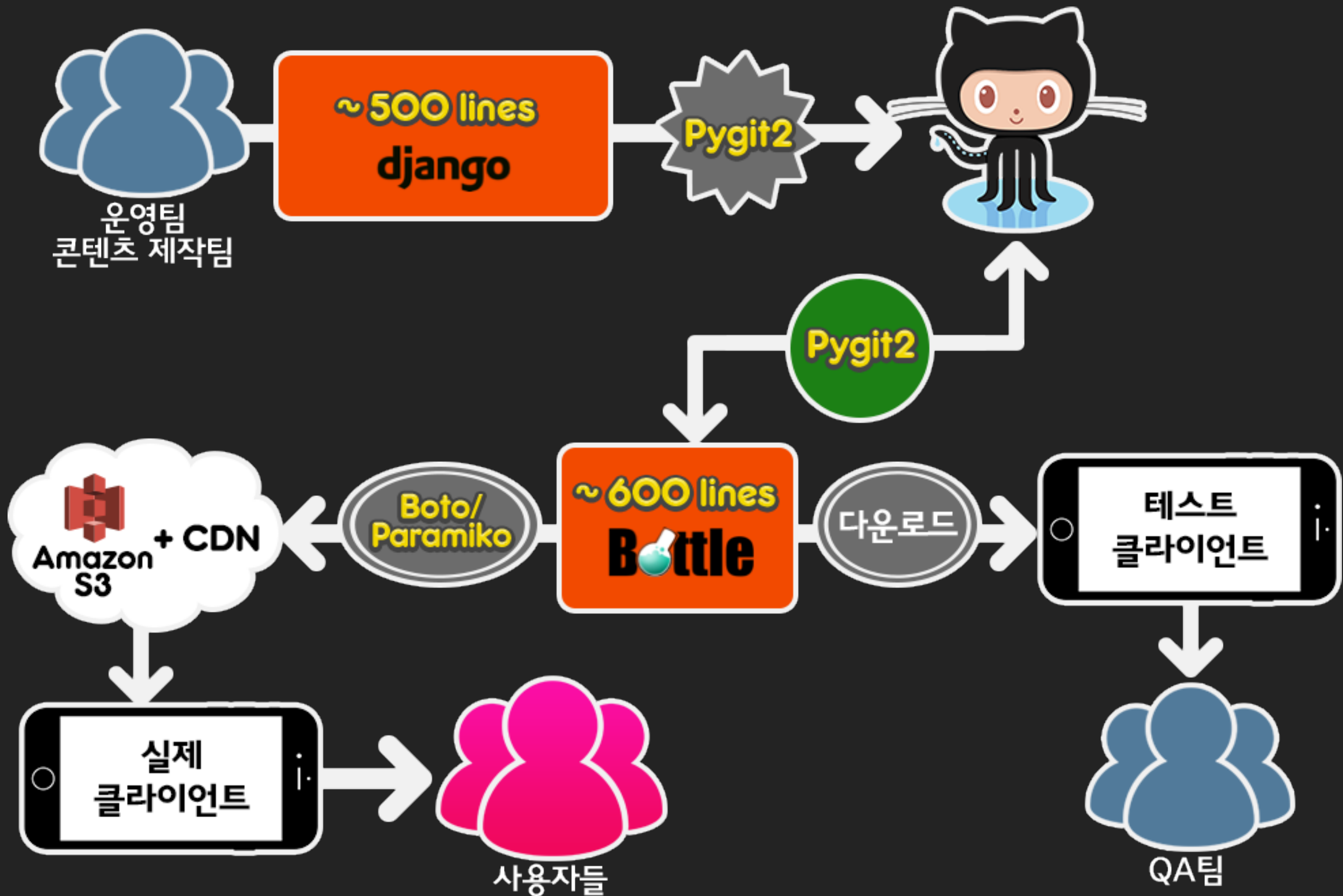
그외에 라이브러리

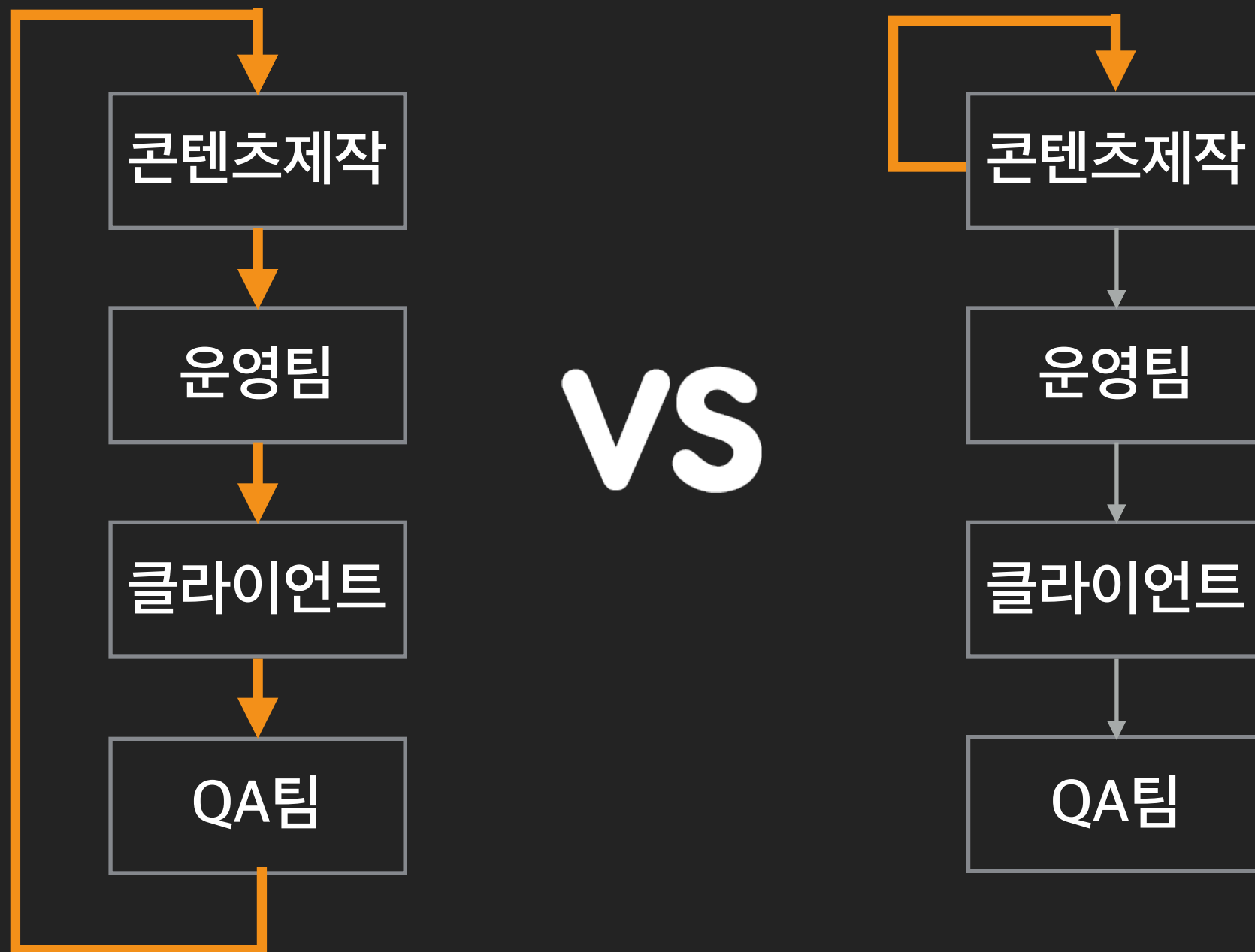
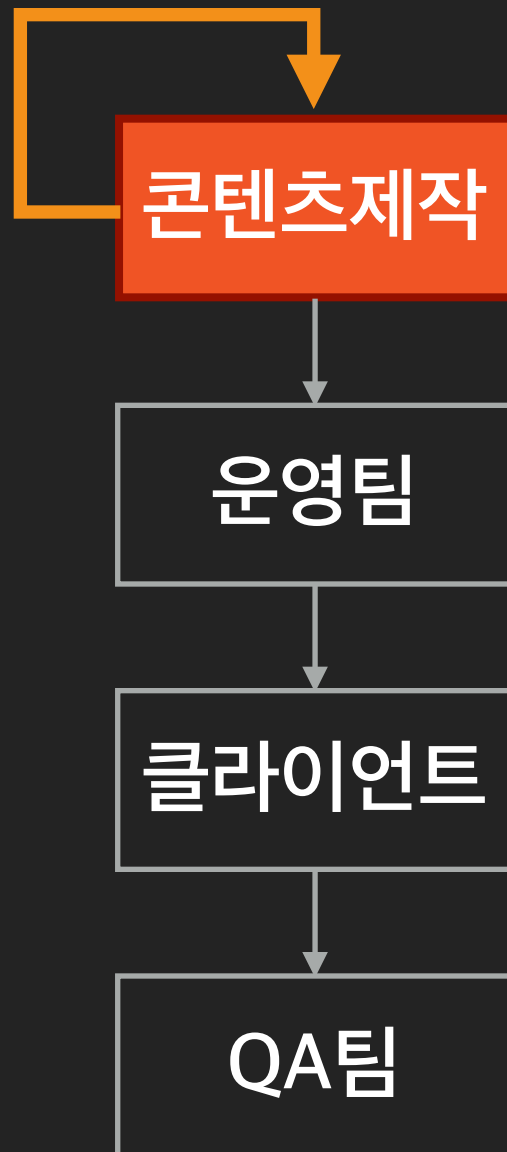
boto - AWS Python SDK

paramiko - SSH2 프로토콜 라이브러리

fabric - 서버 배포에 특화된 SSH 라이브러리

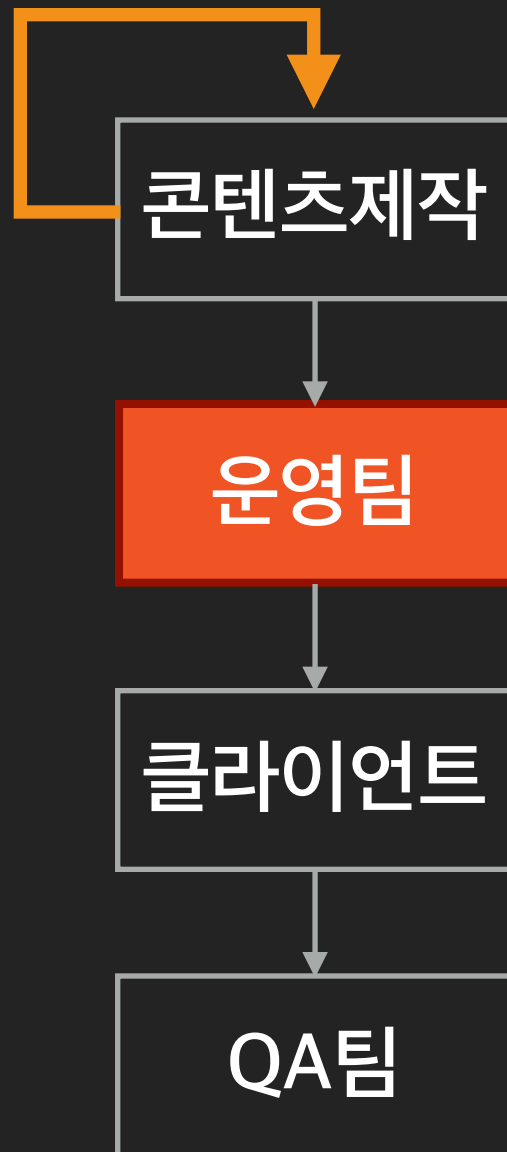
운영팀
콘텐츠 제작팀
Git 웹 클라이언트
django
Pygit2
Pygit2
Amazon S3
+ CDN
Boto/
Paramiko
패치 시스템
Bottle
다운로드
테스트
클라이언트
실제
클라이언트
사용자들
QA팀

~500 lines
django
운영팀
콘텐츠 제작팀
Pygit2
Pygit2
Amazon S3
+ CDN
Boto/
Paramiko
~600 lines
Bottle
다운로드
테스트
클라이언트
실제
클라이언트
사용자들
QA팀

새로운 시스템의 효과

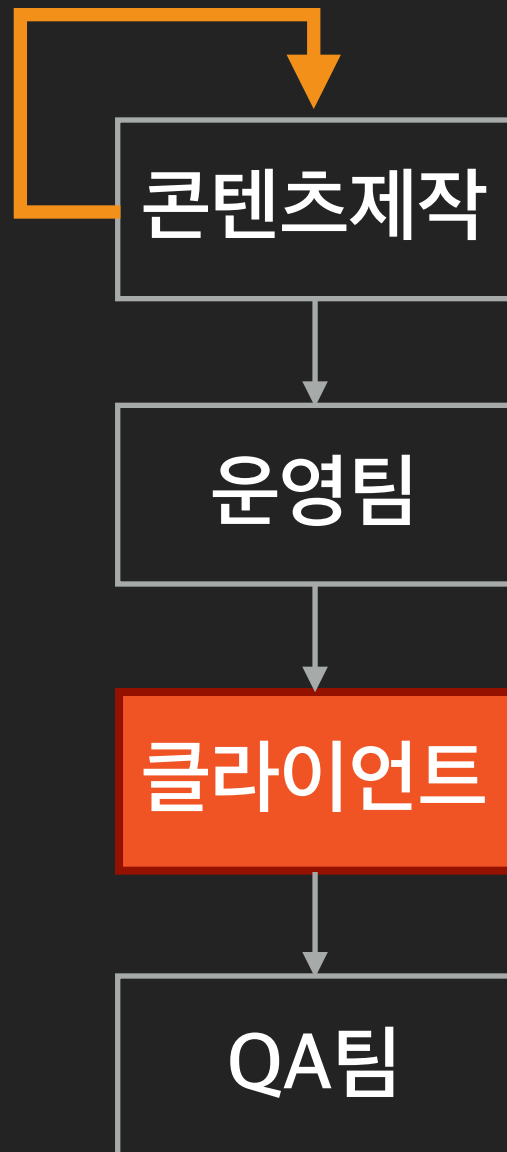
콘텐츠 제작 과정 Before & After

쿠키 굽는 과정 (After)

콘텐츠 확인을 직접 할 수 있게 됨

콘텐츠 확인에 필요한 시간이 매우 단축됨
(수시간 -> 10분 이하)

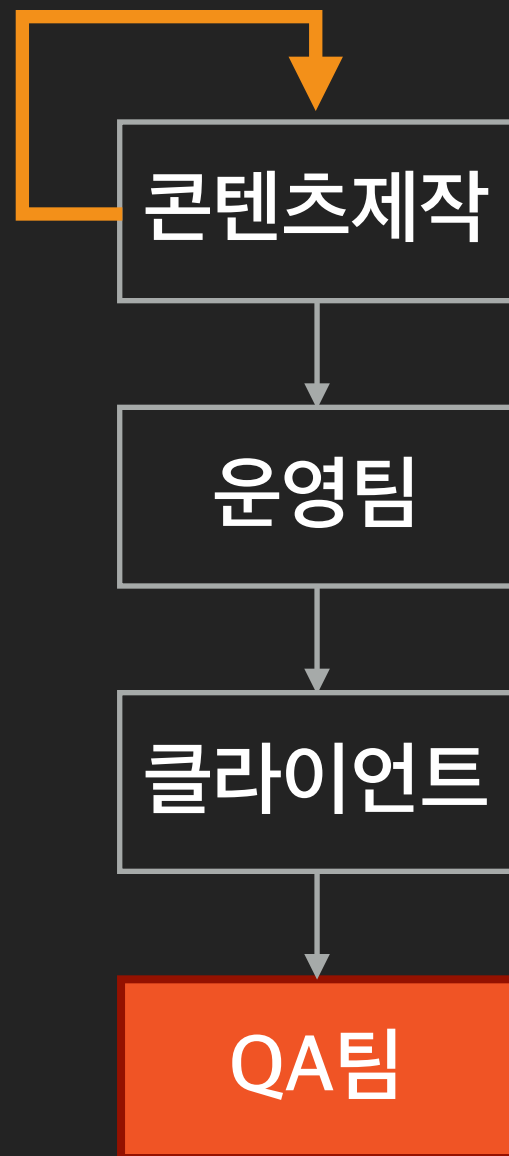
콘텐츠를 자주 확인하고
빠르게 문제를 해결할 수 있게 됨

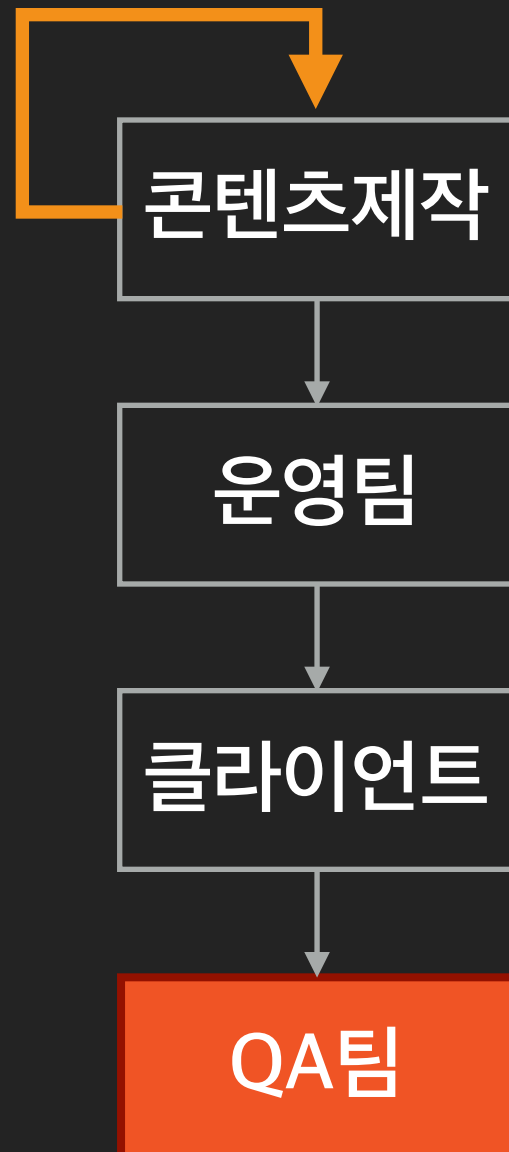
쿠키 굽는 과정 (After)

매 콘텐츠 제작때 발생하는
데이터 반영 요청 감소

배포된 콘텐츠의 이력을 명확히 파악하여
배포상의 실수를 방지하고
발생해도 빠르게 파악

쿠키 굽는 과정 (After)

문제 상황 확인을 위해
빌드를 새로 뽑아야하는 상황 감소

독립적으로 안정화된 콘텐츠 묶음을
이용하여 코드 디버그에 집중

쿠키 굽는 과정 (After)

어쨌든 일이 줄어듦

쿠키 굽는 과정 (After)

어쨌든 일이 줄어듦

회고 & 결론

Git을 시스템 베이스로 사용한 것이 옳았는가?

테스트 코드 부재의 아쉬움

시스템의 Github 의존성

회고 1. 베이스 시스템 Git

Git을 사용한 것이 과연 좋은 선택이었을까?

공통적인 개념을 가져다 쓸 때는 편했다

예측하지 못한 부수적인 개발 공수(삽질)가 많이 들었다

Git도 괜찮았지만, 반드시 Git은 아니어도 된다

회고 2. 테스트 코드 부재

콘텐츠 배포과정에 상시 사용되는
매우 중요한 코드에 테스트가 없다!

문제가 생기면 실제 환경의 유저들 접속이 불가해짐

그런데 테스트가 없어 모르는 사람이 함부로 수정하기 어려움

여러분, 테스트 코드를 짭시다

회고 2. 테스트 코드 부재

이런거 안됩니다

회고 3. Github 의존성

시스템 전체가 Github에 의존적

(원격 저장소, Webhook 등)

대륙발로 추정되는 DDoS로 인해 Github가 작동하지 않았음

Github 장애로 개발자들이 멘붕할 때 콘텐츠 개발팀과 운영팀도 함께

멘붕

별도의 Git 서버를 구축하여 결국은 회피

~~하지만 구축과 동시에 Github가 정상으로 돌아왔…~~

결론

총 배포 회수 200회 이상, 총 전송 트래픽 6 PB 이상

콘텐츠를 만들고, 점검하고, 배포하는 과정에 들이는 시간이 줄어듦

파이썬이 아니었다면
이 정도 규모의 시스템을 만드는데에 얼마나 걸렸을까?

파이썬을 사용한 것은 탁월한 선택이었다

세상을 즐겁게!

개발자가 개발에 집중할 수 있게
필요한 것을 아낌없이 지원해주는 회사

함께 일하는 것이 즐거운 스마트한 사람들

저희들과 함께 재밌는 거 만들지 않으시렵니까 ;p

환영합니다.

최고의 팀에 합류하세요!

Q&A

감사합니다!